The GridLite DREAM: Bringing the Grid to Your Pocket

Nenad Medvidovic and Chris A. Mattmann
Computer Science Department
Viterbi School of Engineering
University of Southern California
{neno,mattmann}@usc.edu

1. Introduction

The emergence of small, mobile, embedded, inexpensive computing platforms (e.g., PDAs, cell phones, GPS receivers) has made computation possible virtually anywhere. In turn, this has opened up countless possibilities for distributed and decentralized collaboration and information sharing among a wide range of individuals and organizations, including engineers, scientists, health and humanitarian workers, emergency response teams, law enforcement agencies, and average citizens. Fleets of mobile devices are, or will soon be, employed in complex scenarios such as land and sea exploration, environment monitoring, traffic management, fire fighting, and damage surveys in times of natural disaster. The software-intensive systems of today are increasingly shaped by their *d*ecentralized, *r*esource-constrained, *e*mbedded, *a*utonomic, and *m*obile (*DREAM*) computing environments.

In parallel with this development, another exciting and promising direction in modern computing has emerged – *the grid*. Grid computing connects dynamic collections of individuals, institutions, and resources to create virtual organizations, which support sharing, discovery, transformation, and distribution of data and computational resources. Distributed workflow, massive parallel computation, and knowledge discovery are only some of the applications of the grid. Grid applications involve large numbers of distributed devices executing large numbers of computational and data components. As such, they require techniques and tools for supporting their design, implementation, and dynamic evolution.

The grid paradigm, however, makes a number of limiting assumptions that curtail its adoption, utility, and deployment in the emerging DREAM computing environments. These assumptions include availability of powerful processors, large amounts of memory, capacious and reliable network links, and stability of software systems deployed on the grid. In this paper, we will argue for a means to combine these two exciting parallel research areas, in short, to use grid computing as a means of constructing software systems in the DREAM environments. Our position is that, in order to address the aforementioned limitations of the grid paradigm, a new type of software solution must be constructed. To that end we propose, and have implemented an early prototype of, *GridLite*, a software architecture-based grid platform suitable for deployment in DREAM environments.[1] The ultimate goal of GridLite would be to extend the reach of the grid all the way to people's "pockets".

2. Background

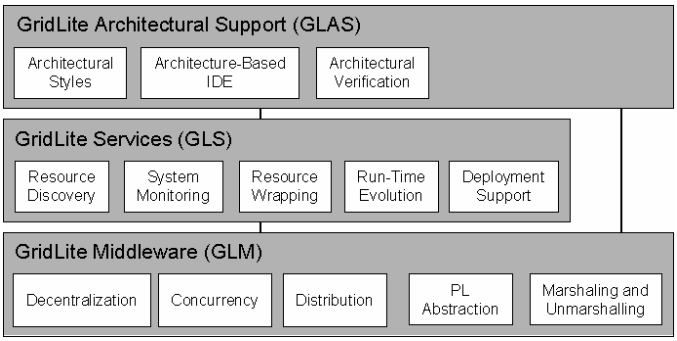
The grid has revolutionized the manner in which both computation-intensive and data-intensive software systems are constructed and deployed. At the same time, two independently conducted studies to date[2,3] have indicated that existing grid technologies suffer from several recurring shortcomings, which are particularly magnified in the context of the emerging DREAM environments. We will highlight several of the shortcomings for illustration. First, existing grid solutions are implemented using technologies (e.g., CORBA, Web services) that are unsuitable for DREAM environments. Second, grid protocols (e.g., Grid Resource Allocation Management, GridFTP, Meta Directory Service) require heavy-weight processing and memory resources to poll and monitor nodes in a grid-based system. Third, the grid solutions assume stable network connectivity and bandwidth. Fourth, the topology of a deployed grid-based system is essentially static, and any modification to an existing deployment may require a (manual) restart of the entire system. Finally, the existing grid technologies provide no system design, implementation, deployment, and evolution guidance to their users; instead, they implicitly assume that the users will somehow "figure it out". This is particularly problematic when one considers that the systems deployed on the grid may

1. C. A. Mattmann, S. Malek, N. Beckman, M. Mikic-Rakic, N. Medvidovic, and D. J. Crichton. GLIDE: A Grid-based Light-weight Infrastructure for Data-intensive Environments. *Proceedings of the 2005 European Grid Conference (EGC 2005)*, Amsterdam, the Netherlands, February 2005.
2. A. Finkelstein, C. Gryce and J. Lewis-Bowen, Relating Requirements and Architectures: A Study of Data-Grids. *Journal of Grid Computing*, vol. 2, pp. 207-222, September 2004.
3. C. A. Mattmann, N. Medvidovic, P. M. Ramirez, and V. Jakobac. Unlocking the Grid. *Proceedings of the 8th International Symposium on Component Based Software Engineering (CBSE-8)*, St. Louis, MO, May 2005.

be highly complex, and that the typical users of the grid (e.g., scientists, health workers) may have no formal training in software development.

While most of the above difficulties may be overcome by engineering more efficient underlying infrastructure, the lack of development guidance for grid-based software systems also requires enriching grid computing with an appropriate body of software development concepts, constructs, principles, and techniques. We believe that the area of software architecture provides such a body of knowledge. Software architectures are high-level abstractions for modeling the structure, behavior, and key properties of software systems. These abstractions involve descriptions of elements from which systems are built, interactions among the elements, patterns that guide their composition, and constraints on those patterns. In general, a system is defined as a set of *components* (elements that encapsulate computations and state in a system), *connectors* (elements that embody interactions), and a *configuration* (overall organization of components and connectors). Furthermore, software architectural *styles* are key design idioms that embody best practices in the design of systems in specific domains (e.g., DREAM). Software architecture will play an important role in our proposed research agenda, as detailed below.

3. Objectives of GridLite



In order to address the above-stated shortcomings of existing grid technologies and bring the grid "to your pocket", a new type of grid must be developed, which we dub "GridLite". GridLite is an adaptable, light-weight grid platform that is equipped with the appropriate support for application design, implementation, deployment, and evolution. To realize GridLite, we have drawn upon our experience in the areas of distributed computing,[4] software architecture and in particular its role in DREAM environments,[5] and distributed data management.[1] The objectives of GridLite are be three-fold: (1) creation of an efficient and adaptable middleware platform to support implementation, deployment, and runtime monitoring and evolution of GridLite systems; (2) development of an extensible set of grid services (e.g. resource discovery, system monitoring, and so forth); and (3) formulation of a set of software architectural principles for constructing GridLite-based applications. We see GridLite as a layered solution comprising facilities for software architecture-based application development that leverage a set of core grid services, both of which are implemented on top of a light-weight middleware platform, as depicted in the above figure.

The proposed middleware layer (GLM) builds on our prior work in the area of light-weight middleware,[3] and encapsulates low-level communication on devices whose connectivity is limited and unstable. GLM is intended to abstract different communication protocols (e.g., Internet, Bluetooth) to enable transparent connections; different operating systems (e.g., PalmOS, WindowsCE, Symbian) to ensure platform portability; and different programming languages (e.g., J2ME, EVC++, Brew) to foster development flexibility and interoperability. GLM should also leverage light-weight software components (e.g., "tiny" XML parsers) to provide application extensibility, resource discovery, and data conversion. Ultimately the GLM layer should provide compatibility with existing grids, extending the reach of grid computing to pocket-sized devices. Design, implementation, and evaluation of GLM has been the early focus of our work on GridLite.

The proposed services layer (GLS) encapsulates the lower-level interfaces of GLM into a set of basic GridLite services. The services should include discovering resources on the grid (such as data or computation provision), monitoring grid and grid node performance, wrapping structured resources to enable access in a heterogeneous setting, dynamically "morphing" deployed system architectures, and deploying uniform grid software components in the face of different execution platforms. Experience in resource and metadata description[1] and software architectures[4] will be necessary to enable GLS to utilize GLM effectively and efficiently.

4. S. Malek, M. Mikic-Rakic, and N. Medvidovic. A Style-Aware Architectural Middleware for Resource-Constrained, Distributed Systems. *IEEE Transactions on Software Engineering*, vol. 31, no. 3, pp. 256-272, March 2005.

5. N. Medvidovic, M. Mikic-Rakic, N. Mehta, and S. Malek. Software Architectural Support for Handheld Computing. *IEEE Computer, Special Issue on Handheld Computing*, vol. 36, no. 9, pp. 66-73, September 2003.

The final layer of GridLite provides software architectural support (GLAS). We believe that a software architecture-based approach is essential in deploying successful applications onto hundreds or thousands of DREAM nodes. GLAS should enable the expression of application architectures using a number of systematic primitives (e.g., component, connector, port, communication event, data stream) that are directly implemented in the GLM. From these primitives, complex architectures could be shaped and then dynamically reshaped based on environment events, changing requirements, dynamically discovered resources, and so on.

4. Relationsip to the Workshop Theme

The 2005 Monterey Workshop focuses on five key challenges in designing software for networked systems: (1) System Integration and Dynamic Adaptation, (2) Effects of Dynamic Structure, (3) Effects of Faults, (4) Design for Reliability, and (5) Effects of Scale.[6] To make GridLite tangible, all five of these research challenges must be addressed, but it is our position that three of the challenges are particularly important. We elaborate on them below.

4.1 System Integration and Dynamic Adaptation

Existing grid-based software implementations are almost entirely disconnected from their architectural models; in many instances, no explicit such models even exist. This disconnection usually leads to *architectural drift*, where a software system's implementation evolves over time while its original architectural model remains static. To address this, a grid system's architecture must remain a "living artifact" and evolve along with its implementation. In particular, we assert that this can be effectively accomplished by providing native implementation-level support for architectural concepts (e.g., in the form of an "architectural middleware").[4] Our early experience building the GLIDE middleware[1] has provided early evidence of the success of applying this approach to the DREAM domain.

4.2 Effects of Faults

Grid-based software implementations assume highly reliable network connectivity and high-bandwidth connections. However, in DREAM environments the opposite is often the case. DREAM systems are increasingly operating in computing environments with frequent network interruptions, and bandwidth capabilities are highly variable. To effectively utilize the benefits of grid computing in DREAM environments, grid systems will have to continue to operate "normally" in the face of such customary disconnections. Also, grid software systems will need to be engineered to make the best use of the bandwidth made available to them. We postulate that this challenge can be overcome most effectively via a combination of low-level networking facilities and higher-level services provided by a middleware technology such as GridLite.

4.3 Effects of Scale

Current grid systems are typically deployed to support scientific research networks, which, though highly geographically distributed (in most cases across continents), are not larger than 5-10 nodes. To deploy grid software systems in DREAM environments that include thousands of nodes (e.g., onto cell phones), the scale of grid software must be significantly increased. Determining effective architectural solutions for addressing this problem is a critical research challenge. We hypothesize that formulating and supporting appropriate architectural styles, as proposed in the GridLite architecture, is a promising direction to pursue.

5. Conclusions

The GridLite vision that we argue for in this paper has the potential to shape the role and utility of existing mobile computing platforms such as PDAs, cell phones, and laptops used in everyday life. As networked computing systems must operate in DREAM environments, several avenues of research must be significantly advanced. In this paper, we have argued that grid computing provides a promising approach for engineering some classes of networked computing systems, but several limitations of the grid must be addressed before its widespread deployment and use as a computing platform for future networked environments.

6. 2005 Monterey Workshop – Objectives. http://www-src.lip6.fr/homepages/Fabrice.Kordon/Monterey/objectives.html